Stage 4 Features
“You need at least one each of transactions, stored procedure, trigger, and constraints. The transaction and stored procedure need to be separate, meaning if you put the transaction in a stored procedure, that only counts for the transaction requirements and you need a separate stored procedure.”

Transaction

Update User Year If 18 Credits

```
CREATE PROCEDURE UpdateUserYearIf18Credits(IN netid VARCHAR(20))
BEGIN
   DECLARE totalCredits INT;

   START TRANSACTION;

   SELECT SUM(c.Credits)
   INTO totalCredits
   FROM ENROLLS e
   JOIN COURSE c ON e.CRN = c.CRN
   WHERE e.NetID = netid;

   UPDATE `USER`
   SET Year = CASE
      WHEN Year = 'Freshman' AND totalCredits >= 18 THEN 'Sophomore'
      WHEN Year = 'Sophomore' AND totalCredits >= 18 THEN 'Junior'
      WHEN Year = 'Junior' AND totalCredits >= 18 THEN 'Senior'
      ELSE Year
   END
   WHERE NetID = netid;

   COMMIT;

END //

DELIMITER;
```

Stored Procedure

Instructor Evaluation Based on Time Spent, Class Difficulty, and Rating

```
DELIMITER //
CREATE PROCEDURE HighLowDifficulty()
BEGIN
   (SELECT r.Name AS InstructorName,
        AVG(r.ClassDifficulty) AS AverageClassDifficulty,
        AVG(r.TimeSpent) AS AverageTimeSpent,
        AVG(r.Rating) AS AverageRating
   FROM RATES r
   GROUP BY r.Name
   HAVING AverageTimeSpent < 5 AND AverageClassDifficulty < 3 AND AverageRating > 3
   LIMIT 10)
   UNION
  (SELECT r.Name as InstructorName,
        AVG(r.ClassDifficulty) AS AverageClassDifficulty,
        AVG(r.TimeSpent) AS AverageTimeSpent,
        AVG(r.Rating) AS AverageRating
  FROM RATES r
  GROUP BY r.Name
   HAVING AverageTimeSpent > 15 AND AverageClassDifficulty > 7 AND AverageRating < 3
   LIMIT 15);

(SELECT d.Title as DepartmentTitle,
        AVG(r.ClassDifficulty) AS AvgClassDifficulty,
        AVG(r.TimeSpent) AS AvgTimeSpent,
        AVG(r.Rating) AS AvgRating
   FROM RATES r NATURAL JOIN INSTRUCTOR i JOIN DEPARTMENT d ON i.Title =
        d.Title
  GROUP BY d.Title
  HAVING AvgClassDifficulty > 7 OR AvgTimeSpent > 15 OR AvgRating < 3
  ORDER BY DepartmentTitle ASC
  LIMIT 5)
UNION
  (SELECT d.Title as DepartmentTitle,
        AVG(r.ClassDifficulty) AS AvgClassDifficulty,
        AVG(r.TimeSpent) AS AvgTimeSpent,
        AVG(r.Rating) AS AvgRating
```

```
    FROM RATES r NATURAL JOIN INSTRUCTOR i JOIN DEPARTMENT d ON i.Title =
         d.Title
   GROUP BY d.Title
   HAVING AvgClassDifficulty < 3 OR AvgTimeSpent < 5 OR AvgRating > 3
   ORDER BY DepartmentTitle ASC
   LIMIT 5);
END //
DELIMITER ;
```

Trigger

Enrollment Credit Limit Check

```
DELIMITER //

CREATE TRIGGER CheckCredits
BEFORE INSERT ON ENROLLS
FOR EACH ROW
BEGIN
   DECLARE current_credits INT;
   DECLARE new_course_credits INT;

   SELECT IFNULL(SUM(c.Credits), 0) INTO current_credits
   FROM ENROLLS e
   JOIN COURSE c ON e.CRN = c.CRN
   WHERE e.NetID = NEW.NetID;

   SELECT Credits INTO new_course_credits
   FROM COURSE
   WHERE CRN = NEW.CRN;

   IF current_credits + new_course_credits > 18 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Exceeding the 18 credit limit is not allowed';
   END IF;
END //

DELIMITER ;
```

Constraints

Foreign key references, for example in the Instructor table:

```
CREATE TABLE Instructor (
   Name VARCHAR(255) PRIMARY KEY,
   Email VARCHAR(255),
   Title VARCHAR(255),
   FOREIGN KEY (Title) REFERENCES Department(Title)
);
```